JSIMS Internal HLA I/F Specification Modifications

JSIMS Enterprise

Dr. David Pratt
Technical Director
prattd@jsims.mil

Presented at the Architecture Management Group
April 9 and 10, 1997

Outline

- What is JSIMS?
- JSIMS challenges relevant to the HLA
- Proposed changes to the HLA RTI Interface Specification and API to meet JSIMS needs

As the HLA evolves, the JSIMS partner programs need to take a active role in shaping the future directions.
This presentation represents the first evaluation of what is needed for **internal compliance**.

JSIMS Warfighter Vision

JSIMS Enterprise

- JSIMS is a simulation system that supports the twenty-first century warfighter's *preparation for real world contingencies*.
- The system provides *garrison and deployed* exercise capability to meet *current and emerging training and operational requirements* in a *timely and efficient* manner.
- By interfacing to the warfighter's real *go to war systems*, the view into the simulation world *mirrors* that of the real world.

JSIMS Technical Vision

JSIMS Enterprise

- JSIMS is a *single, distributed, seamlessly integrated simulation environment*.
- It includes a *core infrastructure* and *mission space objects*, both maintained in a *common repository*.
- These can be *composed* to create a simulation capability to support *Joint or Service* training, rehearsal, or education objectives.

JSIMS Challenges

JSIMS Enterprise

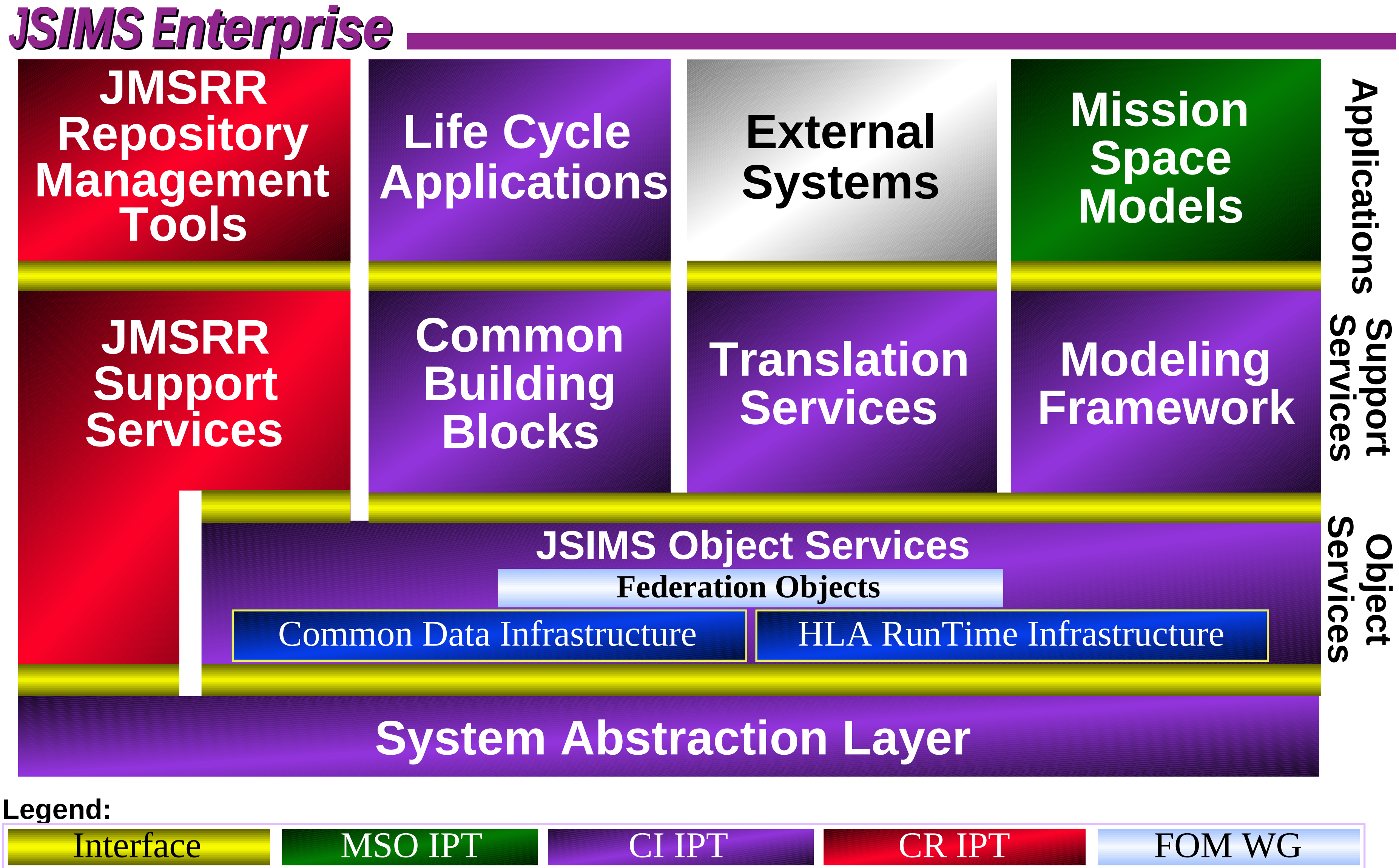
- Composability
- Scalability
- Expandability
- Interoperability
- Reduced exercise overhead
- Reusability
- Security

➢ JSIMS is an ***extreme performance*** federation

- Hundreds of thousands of objects
- Dozens or hundreds of computing hosts
- Real-time performance with Human in the Loop C^4I interfaces
- Real-time, scaled faster than real-time, and as fast as can run performance under a range of conditions (e.g., DSS, doctrine, education)
- Widely distributed
- Multiple Levels/Multi-level Security
- Using many network/communications technologies potentially simultaneously
- Tight coupling with C^4I systems
- Interoperability between many hundreds of independently-developed models

↳ For JSIMS to be internally HLA-compliant and meet these goals, certain additions are needed to the current HLA definition, many of which are implementation vs. definitional issues. Many of the needed enhancements have been demonstrated in the STOW RTI-S version.

JSIMS Layered Software Architecture

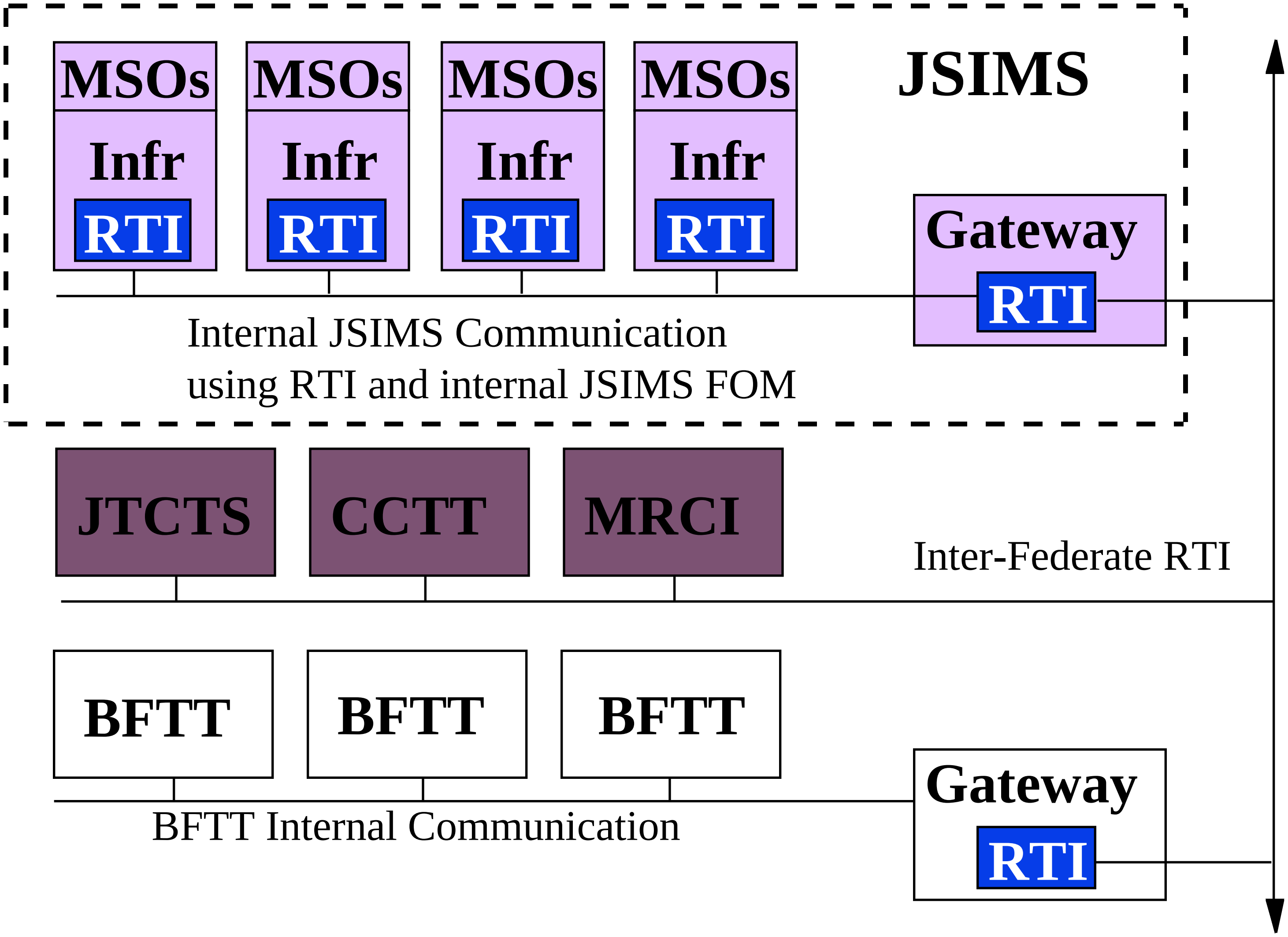


JSIMS Approach to HLA Compliance

JSIMS Enterprise

- The JSIMS Architecture is fully HLA compliant:
  - JSIMS is its own federation
    - need to clarify rules concerning distributed federates
  - The JSIMS Federation Objects are the JSIMS FOM, which will be documented using the HLA OMT format
  - All JSIMS run-time state data is distributed using the RTI
  - All RTI interaction is via the RTI interface specification
  - Each attribute of each federation object is owned by a single model which may change during execution
  - Each model and/or application is a federate and the objects/interactions it supports are its SOM
  - The JSIMS Modeling Framework supports ownership transfer via the RTI
  - JSIMS time management functionality is based solely on the RTI time management facilities, the definition & implementation of which is insufficient given current RTI developments

JSIMS Approach to Federation with other HLA-Compliant Systems



The JSIMS Gateway translates between the JSIMS internal FOM (now considered the JSIMS SOM) and the Common FOM.

Communication between JSIMS and other federates uses the RTI and a commonly agreed upon FOM.

Latency/Bundling

- Recommendation:
  - The application should be able to specify to the RTI the maximum tolerable latency and maximum tolerable packet size.
    - Relative priority of data.
- Rationale:
  - The ability to bundle attributes into packages is considered critical to achieving performance gains for JSIMS.
  - Bundling, however, increases latency.
  - Thus, the application needs to be able to carefully control how latency and packet size are traded off to achieve maximum performance

Network QoS Parameters

- Recommendation:
  - The application needs to be able to specify to the RTI any network Quality-of-Service parameters needed to operate in a particular network environment.
  - If the RTI can detect the performance parameters of the underlying network, it needs to be able to communicate this information back up to the application.
- Rationale
  - In general, the RTI controls multicast groups or point-to-multipoint connections.
  - In some situations, the application must decide on how to apportion resources, thus it needs information from the RTI on underlying network technology, as well as the ability to set network parameters to meet the exercise’s objectives.
- Other comments:
  - Some of this might be done via interactions with the MOM.

Active/passive subscription based on network address

- Recommendation:
  - The application must be able to specify that the RTI subscribe to all data on a particular LAN under both of the following conditions:
    - Subscribe to all information on a particular LAN without causing any more traffic to flow than would in the absence of this subscription. (Passive)
    - Subscribe to all information that is being generated on a *remote* LAN and cause that information to flow to the current machine. (Active)
- Rationale:
  - Data collectors need to collect all information on a given LAN, to allow efficient parallel collection of data under certain circumstances.
  - Sometimes these data collectors are not located on the LAN they want to collect from.

Direct Publication/Subscription to MC Groups (or their abstract equivalent)

- Recommendation:
  - The application needs to be able to publish and subscribe directly to the underlying multicast groups, rather than using the DDM interface.
- Rationale:
  - Large exercises require distributed data collection, in which a number of collector machines sample some fraction of the traffic.
    - If the exercise is large enough, numerous loggers are needed for each LAN.
  - In this case, they need to be "load-balanced," to insure that individual collectors do not get overloaded.
    - Because Routing Spaces do not give the application enough control over what data gets collected, it is impossible to efficiently load balance using Routing Spaces (i.e. the Routing Space "knob" is not precise enough).
    - The only way to efficiently load-balance is to enable subscription based on individual MC groups, since these are the actual underlying mechanisms for data transfer.
  - Playback requires publication to the same groups.

More information is required about data being delivered

- Recommendation:
  - For each *reflect_attribute_values* and r*eflect_interaction* call made by the RTI to the application the RTI needs to provide additional meta-data:
    - what MC Group/Stream/Channel the information came from,
    - the transport type of the data,
    - the time management scheme under which the data was sent,
    - what Routing Space Subscription this data was in response to.
- Rationale:
  - The first three pieces of information are required for accurate playback.
  - The last piece of information is required to efficiently deliver the data to the part/component of the application that asked for it.

The RTI should not read or write directly to the hard disk.

- Recommendation:
  - The RTI API needs to be changed such that the FED and RID information can be delivered to the RTI by the application, rather than having the RTI read directly from the local machine's hard drive.
- Rationale:
  - Allow remote configuration, startup and management of federates.
  - Secure configuration managed systems require all information to be brokered by the appropriate identified and authenticated authority.
  - The easiest way for the RTI to deal with this problem is to push this functionality up into the application, and then read and write to the hard drive through means (i.e. iostreams) provided by the application.
  - The alternative is to build security and configuration management functionality directly into the RTI, making it less "lean and mean."
  - Thus the RTI API needs to be changed to allow the application to deliver and the RTI to request the appropriate iostreams to perform disk I/O.

API additions are needed to tell the RTI to save and restore its state

- Recommendation :
  - Additional API calls are required to tell the RTI to save its state to a user-supplied iostream.
  - Calls are further required to tell the RTI to restore its internal state from a provided iostream.
- Rationale:
  - The RTI, as a piece of the JSIMS system, will need to be able to checkpoint its state and restore it from a checkpoint as part of the the overall system.
  - Even though the RTI does not remember FOM object state, it does have some state information (such as routing tables, subscription lists, etc.) that needs to be preserved from one restart to the next.

The RTI needs the ability to tag data with a security classification

- Recommendation:
  - All data shipped by the RTI needs to be tagged with a security classification provided by the application.
  - Additional “security attributes” will not do the job, as it is difficult to get the RTI to transmit unchanged attributes along with changed attributes.
  - Packets leaving the RTI should contain only information at a single security level (i.e. Unclassified or Secret, etc.) Mixed-classification packets must be disallowed.
- Rationale:
  - Security is a very large issue for JSIMS.
  - JSIMS has a three-phase security plan leading to a multi-level secure system.
  - The change above will allow JSIMS to operate at least initially in a Phase II-Multiple Levels of Security environment.
- Other comments:
  - The security implications for RTI software engineering are only now being addressed in a systematic fashion by the JSIMS Security Working Group. Expect more requirements as the JSIMS security plan evolves.

Proposed Changes/Additions to I/F Spec (Summary)

- Latency/bundling parameters changeable by applications
- Network QoS parameters need to be set by the applications
- Active/passive subscription based on network address
- Publication and subscription directly to MC groups (or their abstract equivalent) as well as current methods
- More information is required about data being delivered
- The RTI must not read or write directly from/to the hard drive
- The RTI must be able to comprehensively save and restore its state
- Data needs to be tagged with security level

Many of these features can be implemented below the I/F Spec and hidden from the user.
The question is should they?

Summary and Conclusions

JSIMS Enterprise

- JSIMS is an extreme performance federation with a large array of difficult-to-satisfy requirements
- JSIMS needs additions to the RTI API and I/F spec to support its current architecture, which is internally HLA-compliant
- Most of the changes JSIMS requires have been pioneered on the STOW program as part of the RTI-s design and development

The JSIMS program, as the DoD's flagship simulation program, wishes to evolve the HLA in a direction consistent with its goals of plug-and-play composability, object-oriented design and implementation from the ground up, multiple levels/multi-level security, enhanced reliability, and reduced exercise overhead. JSIMS looks forward to working with the AMG to achieve these goals.

JSIMS Internal RTI Implementation Issues

JSIMS Enterprise

Dr. David Pratt
Technical Director
prattd@jsims.mil

Presented at the Architecture Management Group
April 9 and 10, 1997

Outline

JSIMS Enterprise

- Suggested approaches to RTI internal architecture and software engineering methods to accommodate JSIMS requirements
- Some implementation issues of interest to JSIMS
- Suggested future directions for the HLA

As the HLA evolves, the JSIMS partner programs need to take a active role in shaping the future directions.
This presentation represents the first evaluation of what is needed for **internal compliance**.

Software Engineering / Architectural Issues
Design Specification

- Plug-and-play RTI that allows federates to “pay-as-they-go” for needed functionality
- RTI that is designed to be composable/tailorable to be able to creatively address JSIMS’s extreme requirements
- Clarification on the RTI’s method of addressing threading issues:
  - single threaded application, single threaded RTI
  - multi-threaded application, single threaded RTI
  - single threaded application, multi-threaded RTI
  - multi-threaded application, multi-threaded RTI

Software Engineering / Architectural Issues Design Specification

JSIMS Enterprise

- Approach to memory management that maximizes performance by minimizing data copying inside the RTI and between the RTI and the application
- Documentation on RTI-to-RTI protocols, including how data is communicated and in what format.
- If reliable multicast is used, documentation of exactly how the scheme works, with particular emphasis on how it scales.

Software Engineering / Architectural Issues System Abstraction Layer

- JSIMS has a requirement to build a system that is portable over a wide range of different platforms.
- JSIMS's architectural feature to meet this goal is to have all access to the operating system or hardware be through a System Abstraction Layer (SAL).
- In particular, JSIMS requires as part of the SAL a "Virtual Network Layer," an encapsulation of the underlying communications functionality so that different technologies can be inserted easily into the system:
  - TCP/IP, native ATM, native FDDI, Tactical Packet Network, shared memory

Software Engineering / Architectural Issues System Abstraction Layer

- JSIMS understands that such an encapsulation does not come without cost; however, JSIMS requires that appropriate engineering studies be done to determine the cost of such a layer consistent with maximizing the RTI's performance and delivering a tailorable RTI.
- JSIMS would like to work with the AMG and the RTI 2.0 designers to develop a SAL that can meet both the RTI's and the JSIMS applications' needs

Software Engineering / Architectural Issues
Other Issues

JSIMS Enterprise

- An application using a single instance of the RTI should be able to join and actively participate in multiple federations with multiple FEDs and RIDs at the same time
  - Used for multi-exercise Exercise Management tools required in the JSIMS family of TRDs.
- The RTI must be able to send data to multiple networks simultaneously (at the direction of the application).
  - Used for creating an intelligent Router, routing information from one network (i.e. Ethernet) to another (i.e ATM).

Software Engineering / Architectural Issues
Other Issues

- The RTI should be designed such that the failure of any single federate or group of federates does not cause the entire federation to fail.
  - In such cases, the disposition of objects owned by the failed federate(s) should be specifiable on a federation execution by federation execution basis.
    - hand them off to willing recipients
    - RTI assumes (temporary) ownership
    - they all disappear, etc.
- The RTI should be able to save and restore its state for checkpoint/restart purposes.

RTI Implementation Issues Relevant to JSIMS

JSIMS Enterprise

- TSO should maintain causality and result in a repeatable federation execution.
- JSIMS needs to be able to switch time management schemes in the middle of a federation execution.
- Zero-Lookahead and discrete event timing schemes are required as one of the “modes” of time management.
  - pseudo-real time (including scaled pseudo-real time)
  - event based
- JSIMS needs the RTI to run on many platforms, but in particular:
  - Sun/Solaris, SGI/Irix, PC/NT, PC/Solaris, HP/HPUX, ALPHA/NT.
- Source code needs to be made available for developers.
- Clearly delineated and documented name space conventions.

RTI Implementation Issues Relevant to JSIMS

JSIMS Enterprise

- JSIMS may want an RTI that runs entirely on the Java Virtual Machine in addition to a C++ and Ada version.
- Internal RTI performance information (including updates/reflects per second, routing space efficiency, etc.) should be made available to the application.
- The RTI should not read directly from or write directly to the hard disk, rather to application-provided iostreams.
- Location independent file system names
  - Use of relative path names only

Future Directions For the HLA based on JSIMS Full Operating Capability (FOC) Goals

- Support for the ability to define “Sub-federations” with “Sub-FOMs” used for hierarchically segmenting exercises.
- Complex (“nested”) attributes for arbitrary composition hierarchies.
- Flexible interaction adjudication (receiver, sender, 3rd party, other)
- RTI should automatically handle all data marshaling.
- FOM tools based on object-oriented computer assisted software engineering tools commonly available in the commercial market, along with a more OO approach to FOM definition (i.e. a focus on interfaces and substitutability rather than data formats).

Future Directions For the HLA based on JSIMS Full Operating Capability (FOC) Goals

- Fully Object-oriented API
  - JSIMS is preparing a strawman.
- Security
  - Compile-time type safety through a compiled-in FOM.
  - JSIMS wants the AMG to tackle the tough security issues facing us all, by designing the appropriate amount of security into the RTI.
  - The AMG should then work to get these RTI security features and functions accredited by the appropriate government agency.
- An object-oriented CCSIL (CCSIL 2.0 ?) to support the JSIMS framework-based architecture.

Summary and Conclusions

JSIMS Enterprise

- JSIMS is an <u>extreme performance federation</u> with a large array of difficult-to-satisfy requirements
- JSIMS has a number of architecture, design, and implementation constraints on any given RTI implementation to meet JSIMS performance, security, reliability, and composability requirements

The JSIMS program, as the DoD’s flagship simulation program, wishes to evolve the HLA in a direction consistent with its goals of plug-and-play composability, object-oriented design and implementation from the ground up, multiple levels/multi-level security, enhanced reliability, and reduced exercise overhead. JSIMS looks forward to working with the AMG to achieve these goals.